# CMV_Models

This Rmarkdown contains the models with descriptions used for the CMV primary episode analysis. The create_scripts.sh script converts this into the R script used in the analysis.

There are the following notational descrepancies between the models here and notation used in the manuscript:

1. K = the initial susceptible cell population, denoted $S_0$ in the manuscript.
2. I0 = The latently infected cell, denoted $L$ in the manuscript. Not to be confused with $I_0$, the initial infected cell population.
3. T or Tcell = immune effector compartments, denoted $E$ in the manuscript. Changed because Tcell is too specific.
4. death = death rate of immune effectors, denoted $\gamma$ in the manuscript.

AUC_data is from legacy code for a different implementation of immune response. It is effectively equivalent to setting 'death' ( $\gamma$ in the manuscript) to 0.

## Models

### 1. CMVModel_latent_linear

Basic SIV with linear growth of S

$$\frac{\mathrm{d}S}{\mathrm{d}t} = \lambda - \mu S - \beta SV$$
$$\frac{\mathrm{d}I0}{\mathrm{d}t} = \beta SV - \alpha I0 - \mu I0$$
$$\frac{\mathrm{d}I}{\mathrm{d}t} = \alpha I0 - \delta I$$
$$\frac{\mathrm{d}V}{\mathrm{d}t} = pI - cV$$

```
### This script was created by model_code_documentation/create_scripts.sh
### The following models are described in model_code_documentation/CMV_models.Rmd with corresponding pd

CMVModel_latent_linear = function(t, x, parms, AUCData = NULL){
  with(as.list(c(parms, x)), {

    if(as.logical(is.na(parms["lambda"]))) lambda = K * mu

    dS <- lambda - mu * S - beta * S * V
    dI0 <- beta * S * V - alpha * I0 - mu * I0
    dI <- alpha * I0  -  delta * I
    dV <- p * I - c * V

    res <- c(dS, dI0, dI, dV)
    list(res)
  })
}
```

## 2. CMVModel_latent_immunity_CTL

Additional immunity compartment activated by level of infected cells. Immunity targets I (infected cells)

$$\frac{\mathrm{d}S}{\mathrm{d}t} = \lambda - \mu S - \beta SV$$
$$\frac{\mathrm{d}I0}{\mathrm{d}t} = \beta SV - \alpha I0 - \mu I0$$
$$\frac{\mathrm{d}I}{\mathrm{d}t} = \alpha I0 - \delta I - kIT$$
$$\frac{\mathrm{d}V}{\mathrm{d}t} = pI - cV$$
$$\frac{\mathrm{d}T}{\mathrm{d}t} = \theta \frac{I}{K_I + I} - death * T$$

```
CMVModel_latent_immunity_CTL = function(t, x, parms, AUCData = NULL){
  with(as.list(c(parms, x)), {
    if(as.logical(is.na(parms["lambda"]))) lambda = K * mu

    dS <- lambda - mu * S - beta * S * V
    dI0 <- beta * S * V - alpha * I0 - mu * I0
    dI <- alpha * I0  -  delta * I - k * I * Tcell
    dV <- p * I - c * V
    dT <- theta * (I/(KI+I)) - death * Tcell

    res <- c(dS, dI0, dI, dV, dT)
    list(res)
  })
}
```

## 3. CMVModel_latent_immunity_V

Additional immunity compartment activated by level of virus. Immunity targets V (virus). This model was proposed but the CTL model was considered a better match for the biology. Given viral load data alone, this model cannot be distinguised from the CTL model without specific parameter constraints on the immune compartment parameters that were not known.

$$\frac{\mathrm{d}S}{\mathrm{d}t} = \lambda - \mu S - \beta SV$$
$$\frac{\mathrm{d}I0}{\mathrm{d}t} = \beta SV - \alpha I0 - \mu I0$$
$$\frac{\mathrm{d}I}{\mathrm{d}t} = \alpha I0 - \delta I$$
$$\frac{\mathrm{d}V}{\mathrm{d}t} = pI - cV - kTV$$
$$\frac{\mathrm{d}T}{\mathrm{d}t} = \theta \frac{V}{K_T + V} - death * T$$

```
CMVModel_latent_immunity_V = function(t, x, parms, AUCData = NULL){
  with(as.list(c(parms, x)), {
    if(as.logical(is.na(parms["lambda"]))) lambda = K * mu

    dS <- lambda - mu * S - beta * S * V
    dI0 <- beta * S * V - alpha * I0 - mu * I0
    dI <- alpha * I0  -  delta * I
    dV <- p * I - c * V - k * V * Tcell
    dT <- theta * (V/(KT+V)) - death * Tcell

    res <- c(dS, dI0, dI, dV, dT)
    list(res)
  })
}
```

# Functions

## find_peak

Given a subject's data, will return the time of first value within the window (in logs) of the peak. So window = 1 means first point within a log of the peak. Window = 0 returns the peak time. This is used to fit the expansion slope to CMVModel_latent_linear and calculate $R_0$.

```
find_peak = function(data, window = 1){
  #window is the log range around the peak, 0 would be peak

  peak = max(data$count)

  peak_range = which((peak - data$count) <= window) #find counts within window of measured peak
  peak_day = data$days2[min(peak_range)]
  return(peak_day)
}
```